UG0787
User Guide
PolarFire FPGA Block Flow

NOTE: PDF files are intended to be viewed on the printed page; links and cross-references in this PDF file may point to external files and generate an error when clicked. **View the online help included with software to enable all linked content.**

Microsemi®

Power Matters.™

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Fax: +1 (949) 215-4996
Email:
sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions; security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at **www.microsemi.com.**

5-02-00787-2/08.18

Table of Contents

Block Flow - Overview

Block Flow is a bottom-up design methodology that enables you to use design blocks (“components” in generic terms) as building blocks for your top-level design. These building blocks may have already completed layout and been optimized for timing and power performance for a specific Microsemi device. Using these blocks as part of your top-level design can cut down design time as well as improve timing and power performance. Block advantages include:

- Focus on the timing of critical blocks and ensure the timing across the blocks meets requirements before proceeding to integrate your blocks at the top level.
- Changes in other blocks have no impact on your own block; you can re-use your block without re-optimizing for timing closure.
- The block can be re-used in multiple designs.
- Shorter verification time; you need to re-verify only the portion of the design that has changed.

Block Features

- A Block can be synthesized, simulated, and placed-and-routed the same way as a regular design.
- You can lock the placement and routing of the Block to ensure repeatable performance.
- Performance, placement and routing can be fixed absolutely; however these rules can be relaxed gradually, if necessary, to ensure that you can integrate the Block into your top level project.

Use blocks when:

- You have multiple team members working on different parts of the same design.
- The design is congested (uses 90% or more of the resources on a given die).
- You have difficulty meeting timing by doing the design in its entirety. Blocks enable you to compartmentalize the design and optimize sections before you optimize the entire design.
- You want to re-use some elements of your design.
- You want to use the identical elements multiple times in a single design.
- You want to make small changes in your design and expect to keep most of the design unchanged with guaranteed performance.

You cannot use Blocks with all families, they are family and die specific; if your Block has I/Os it is also package specific.

HDL Supported

- Verilog
- VHDL

Synthesis Tools Supported

- Synplify Pro

Nested Blocks

Nested blocks (blocks instantiated inside other blocks) are supported. When publishing, only one file will be published that contains all the required information (including the nested block).

Creating Blocks - Options and Settings

To enable Block Creation for a new project, from the **Project** menu, choose **New Project**. Check the **Enable Block Creation** checkbox.

In an existing project, from the **Project** menu, choose **Project Settings**. Click **Design Flow** and check the **Enable Block Creation** checkbox.

Synthesis Tool Settings

In Synplify Pro, the I/O Insertion option is disabled when the block is synthesized. Libero automatically disables I/O insertion for you before invoking Synplify Pro.

Synthesis

During Synthesis, Libero SoC software adds BLOCK_INTERFACE_I* instances to the block. These instances are virtual buffers added to:

- Improve timing values for the block.
- Provide you with a clear interface to floorplan
- Help with clustering constraints

The BLOCK_INTERFACE_I* instances are removed when the block is published.

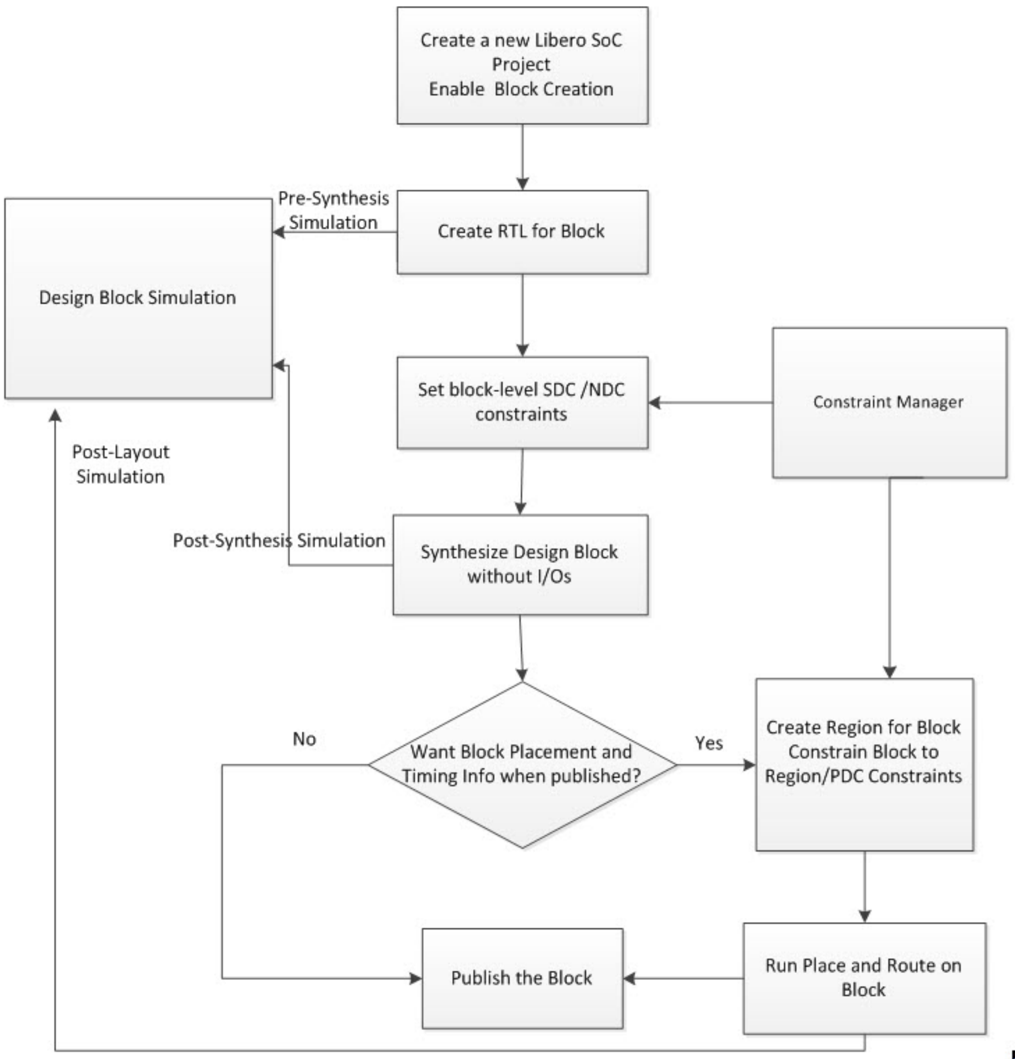

Create a new Libero SoC Project
Enable Block Creation
Create RTL for Block
Pre-Synthesis Simulation
Design Block Simulation
Set block-level SDC /NDC constraints
Constraint Manager
Post-Layout Simulation
Post-Synthesis Simulation
Synthesize Design Block without I/Os
No
Want Block Placement and Timing Info when published?
Yes
Create Region for Block Constrain Block to Region/PDC Constraints
Publish the Block
Run Place and Route on Block

Figure 1 · Creating and Publishing Design Blocks

Publish Options/Settings

Use the Publish Block – Configuration Options dialog box to configure the block for Publication.

Publishing Blocks After Synthesis or Layout

You can publish your block after Synthesis or Layout.

Publish After Synthesis

If you publish a block after Synthesis but before Layout, a netlist is exported for the block when published. No Place and Route information or Region Constraint information is included in the block when published. A Warning message appears when you publish a block before Place and Route.

Publish After Layout

If you publish a block after Layout, the Placement, Routing and/or Region Constraint information will be published along with the netlist. You can always open the configurator and change the options to publish what you want. All macros must be locked or assigned to regions in order to publish the Placement information.

Published Content

Libero exports the <design>.cxz file to <project folder>/designer/<design_block_name>/export folder when a block is published. The <design>.cxz file is a zip file that contains the following files:

- <design_block_name>_syn.v | <design_block_name>_syn.vhd -- A timing shell file passed to synthesis tools when the top-level design is synthesized. The block is marked and treated as a black box when the top-level design is synthesized.
- <design_block_name>_sim.v | <design_block_name>_sim.vhd -- A structural HDL netlist for postsynthesis simulation of the block.
- header_report.log - A log file that contains Header Information on what and how a block is published , including the options you selected to configure the publication.
- <design_block_name>_compile_netlist_resources.xml -- Compile Report detailing resource usage, device info, and a list of high-fanout nets.
- <design_block_name>_gp_report.xml -- Global Placement and Routing Report
- <design_block_name>_compile_netlist_combinational_loops.xml -- Combinational Loops Report
- <design>.cdb – Internal proprietary file containing the optimized netlist , placement, routing or timing constraint information
- <design_block_name>.sdc - contains the SDC constraints for the block to be used for Timing Verifications.

The <design_block_name>.cxz file is your published block. You can move it to another folder, transfer it to other team members, etc. This is the file you import into your top-level design when you want to instantiate the block.

Guidelines for Creating Blocks

Macros/IPs Not Supported in Blocks

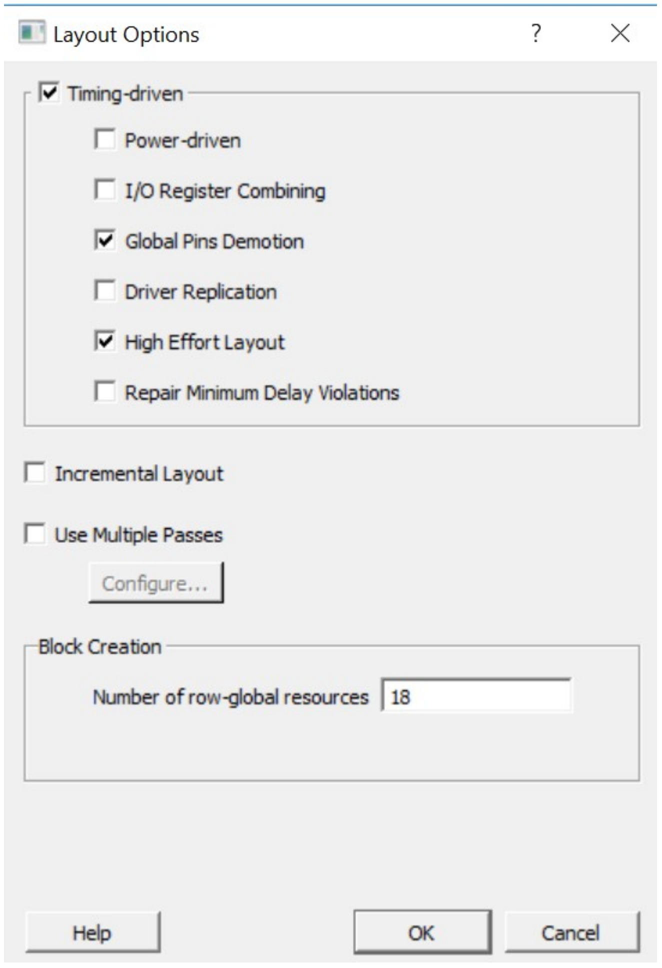
When creating a Block for instantiation in a top-level design, please note that the following types of macros are not allowed: TBD.

Synthesis Tool and Globals Management

The synthesis tool may promote all clocks to globals. Keep in mind the number of globals you need in your top level design and the number of globals allowed in the device (8 or 16 depending on device size) you are targeting. You may need to reduce or limit the number of globals in your block by adding row globals or plan to share globals in the top-level design with the block. To add a row global, you can add it directly to your HDL (RCLKINT).

Place and Route and Globals Management

The Place and Route tool has an option for you to limit the number of row-globals used for block creation.


Layout Options

- [✓] Timing-driven
  - [ ] Power-driven
  - [ ] I/O Register Combining
  - [✓] Global Pins Demotion
  - [ ] Driver Replication
  - [✓] High Effort Layout
  - [ ] Repair Minimum Delay Violations
- [ ] Incremental Layout
- [ ] Use Multiple Passes

Block Creation

Number of row-global resources: 18

Figure 2 · Place and Route Block Creation Option

This option is available only when the Block Creation option is turned on (**Project > Project Settings > Design Flow > Enable Block Creation**).

To use this option, from the Design Flow window, right-click **Place and Route** and choose **Configure Options**. The Layout Options dialog box appears and displays the default number of row-global resources for the technology family. Enter a value to restrict the number of row-global resources available in every half-row of the device. During Place and Route of the block, the tool will not exceed this capacity on any half-row. The default value displayed is the maximum number of row-globals. If you enter a value lower than the maximum capacity (the default), the layout of the block will be able to integrate with the rest of the design if the remaining row-global capacity is consumed.

Blocks and DRC

Regular DRC rules are applied to blocks as in the regular Libero design flow. For example, some DRC rules assume that some pins must be connected to the power nets. These rules are enforced on the blocks in the block flow just as in the regular design flow.

Blocks and Floorplanning

When creating a block, floorplanning is essential if you plan to publish placement information. Before running Layout on the block, you must floorplan the design block. You can use Chip Planner or PDC commands for floorplanning.

If you do not create a region and constrain the Block to the region (floorplanning) or lock the macros before place and route, a Warning message appears when you publish the Block. It warns you that not all macros in the Block have been constrained to regions or locked and therefore only your design netlist is exported when the Block is published.

Floorplanning with PDC Commands

You can use the define_region PDC command to create a rectangular or rectilinear region, and then use the assign_region PDC command to constrain all the macros to that region.

Floorplanning reduces the risk of placement conflicts of the blocks at the top level.

If you do not constrain your Block placement, its components may be placed anywhere on the die.

It is also important to consider the placement of all Block Interface Instances at the boundaries of Block regions. This facilitates the interconnection of the Block to the top-level design. If the Block is highly optimized (densely packed) there may be no routing channels available to connect to any internal Block Interface Instances. Placing all interfaces at Block boundaries helps you eliminate routing congestion and failure.

Floorplanning with Chip Planner

Refer to Chapter 5 of Chip Planner for details on how to use the Chip Planner for floorplanning.

Architecture Limitations - Managing Blocks and Globals

Architecturally, the silicon has 8 or 16 globals per device, depending on the device size. If you create a block for use in a top level design and you know that the top level design will use close to the maximum number of Globals for the device, it is good practice to minimize the number of Globals when you create the block. Examine the Global Report to see how many Globals have been used for the block. To reduce the number of Globals used in the block, you may consider clock sharing and the use of Row Globals for the block.

To add an internal global on a port, you can use either the Synplify constraints editor (SCOPE) or an SDC file.

For example, to add a CLKINT after a CLK port, the command is:

```
define_attribute {n:CLK} syn_insert_buffer {CLKINT}
```

You may instantiate multiple instances of the same block or multiple blocks in the top-level design.

Microsemi recommends that you create a new project for your top-level design. To do so:

1. From the **Project** menu choose **New Project**.

2. Deselect the **Enable Designer Block Creation** checkbox.
3. Choose the **Family/Die/Package** for the new project for the top-level as follows :

- If the block is a Netlist only and was not published with place and route information, choose the same **Family** as the block for the new project. Choose any Die and Package.
- If the block contains placement information, choose the same **Family** and **Die** as the block for the new project, and choose any Package.
- If the Netlist contains I/O and Placement Information, choose the same **Family**, **Die** and **Package** as the block for the new project.

Import the Block

1. From the **File** menu choose **Import > Blocks**.
2. Browse to the directory that contains your <design_block_name>.cxz file and select it.
3. Click **Open**.

<design_block_name> is imported into the top_level project. Version control is not supported for imported blocks. If you import the same block twice, the existing block is overwritten by the new one.

The files will be imported under <design>\component\work\<design_block_name>.

Review the files in the above directory to view Block Reports.

Create a Top Level Design that Uses Blocks

Use SmartDesign or HDL to create your top level design. If you use HDL you can create HDL for the top level or import a top-level HDL file.

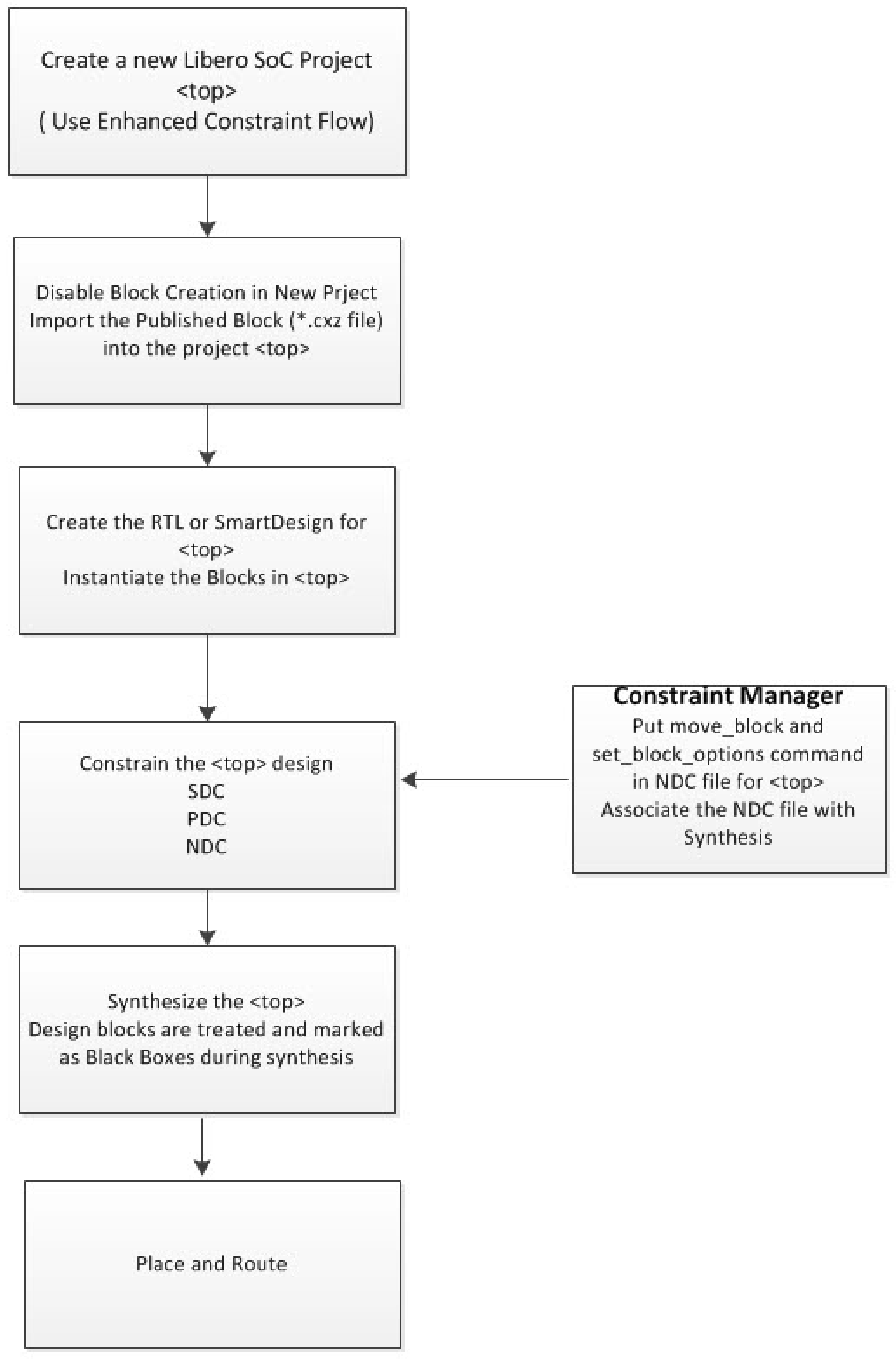

Create a new Libero SoC Project
<top>
(Use Enhanced Constraint Flow)
Disable Block Creation in New Prject
Import the Published Block (*.cxz file) into the project <top>
Create the RTL or SmartDesign for <top>
Instantiate the Blocks in <top>
Constraint Manager
Put move_block and set_block_options command in NDC file for <top>
Associate the NDC file with Synthesis
Constrain the <top> design
SDC
PDC
NDC
Synthesize the <top>
Design blocks are treated and marked as Black Boxes during synthesis
Place and Route

Figure 3 · Instantiating Blocks in your Top Level Design

Constraints Management

When a block with PDC constraints are imported into the top level design, the block's PDC constraints are captured and stored in two files:

- <top_level_module>.block.io.pdc for the IO PDC constraints
- <top_level_module>.block.fp.pdc for the floorplanning PDC constraints.

The <top_level_module>.block.io.pdc is displayed in the I/O Attributes tab of the **Constraint Manager** on top of any other IO PDC files.

The <top_level_module>.block.fp.pdc is displayed in the Floor Planning tab of the **Constraint Manager** on top of any other floorplanning PDC files.

See the Libero SoC Constraint Management chapter of PolarFire FPGA Design Flow User Guide for more information.

Note: Do not modify these block PDC files at the top level. If these PDC files need to be modified, go back to the project where the blocks are created and published. Make the floorplanning modifications and publish the block. Re-import the block into the top level. You may need to remove any duplicate blocks, if any, at the top level after the re-import.

Hierarchical Structure Resolution in Top Level Projects

If you import multiple conflicting definitions for your *.v files, Libero resolves the conflicts as shown below.

Duplicate Block Definition

If you import two versions of your block file you must choose which one you want to use. For example:

1. Import top.v and block1.v files as HDL (**File > Import HDL Source Files**) into the top level project.
2. Import <block1> (**File > Import > Blocks**).

Libero recognizes a duplicate definition of <block1>; one from the HDL and another in the imported block file. The Design Hierarchy tab shows a <block1>.cxf and <block1>.v file under Duplicate Modules; Libero uses the HDL <block1> by default.

To override the default behavior and select the Block definition, right-click the <block1>.cxf file and choose **Use This File**. When you update the behavior the Block icon appears in the Design Hierarchy.

Conflicting Definitions in top.v and Your Imported Block File

You can introduce a conflict if you import a top.v file and a block file. Libero does not support HDL definition of low level blocks inside top level HDL files and subsequent importing of block files. For example, the following will cause an error:

1. Import a top.v file (**File > Import HDL Source Files**) that contains a definition for <top> and a module definition for <block1>.
2. Import the block <block1> (**File > Import > Blocks**).

Libero passes two duplicate files to your synthesis tool because the definition for <block1> is duplicated. To continue, you must remove the definition of <block1> from top.v and then re-import it.

Resolving top.v and Block Instantiations

Libero integrates your top.v file and block file if there is no definition for the block file in top.v. For example:

1. Import your top.v (**File > Import HDL Sources Files**) that contains instantiations but no definition of <block1>.
2. Import <block1> (**File > Import> Blocks**).

Libero resolves the hierarchy for you and puts <block1> under top.v.

EDIF Netlist in the Top Level Design

If the Top Level design is in EDIF, you must convert the EDIF to HDL and then import the HDL into Libero. To convert the Top Level EDIF to HDL:

1. Write a Tcl script. For example:

```
set_device -fam PolarFire
read_edif -file {E:\top.edn}
write_verilog -file {E:\top.v} –skip_empty_modules 1
write_vhdl -file {E:\top.vhd}
## -skip_empty_modules 1 is to instruct the tool not to insert module ## definition for
the empty modules in the HDL created.
```

2. From the Windows Command Prompt or the Linux shell, run rwnetlist as follows (this executable is located in the same location as Libero):

```
rwnetlist --script "E:/run_export_netlist.tcl”
```

Synthesis

Libero passes the block timing to your synthesis tool when the top level is synthesized. This timing shell enables the synthesis tool to produce more accurate timing numbers for top level synthesis.

The timing shell also instructs the synthesis tool to treat the design block as a black box; this is done automatically - no action is required.

Use the Synthesis tool options (**Design Flow > Synthesize > Configure Options**) for "Resolving Place and Route Conflicts" on page 17 of blocks.

Resolving Place and Route Conflicts

To resolve Place and Route conflicts at the top-level:

- Examine the <design_block_name>_compile_netlist_resources.xml Report. Identify the cause of the problem and manually place and constrain the placement with Chip Planner or with PDC commands.
- If you instantiate a block (published with placement) multiple times then placement between multiple block instances will overlap. To remove overlapping, move the block placement of one or more instances to another area using the PDC command move_block. Put the move_block command inside the NDC file and associate the NDC file with Synthesis (**Constraint Manager> Netlist Attributes**)
- The software enforces Global sharing. If there is a Global driving a CLKINT in the block it will be deleted. Reduce the number of Globals at the top level by sharing Global Clock resources. Globals in the Blocks may also be re-routed (not preserved).

Synthesis Options to Resolve Place and Route Conflicts

If there are multiple blocks instantiated in your top level design, the software uses the Synthesis Options to resolve the conflicts. These options appear only if there are blocks in your design. Use the synthesis options (Design Flow > Synthesize > Configure Options) to resolve Placement and/or Routing conflicts.

Placement

Error if conflict - The Layout tool errors out if any instance from a designer block is unplaced. This is the default option.

Resolve conflict

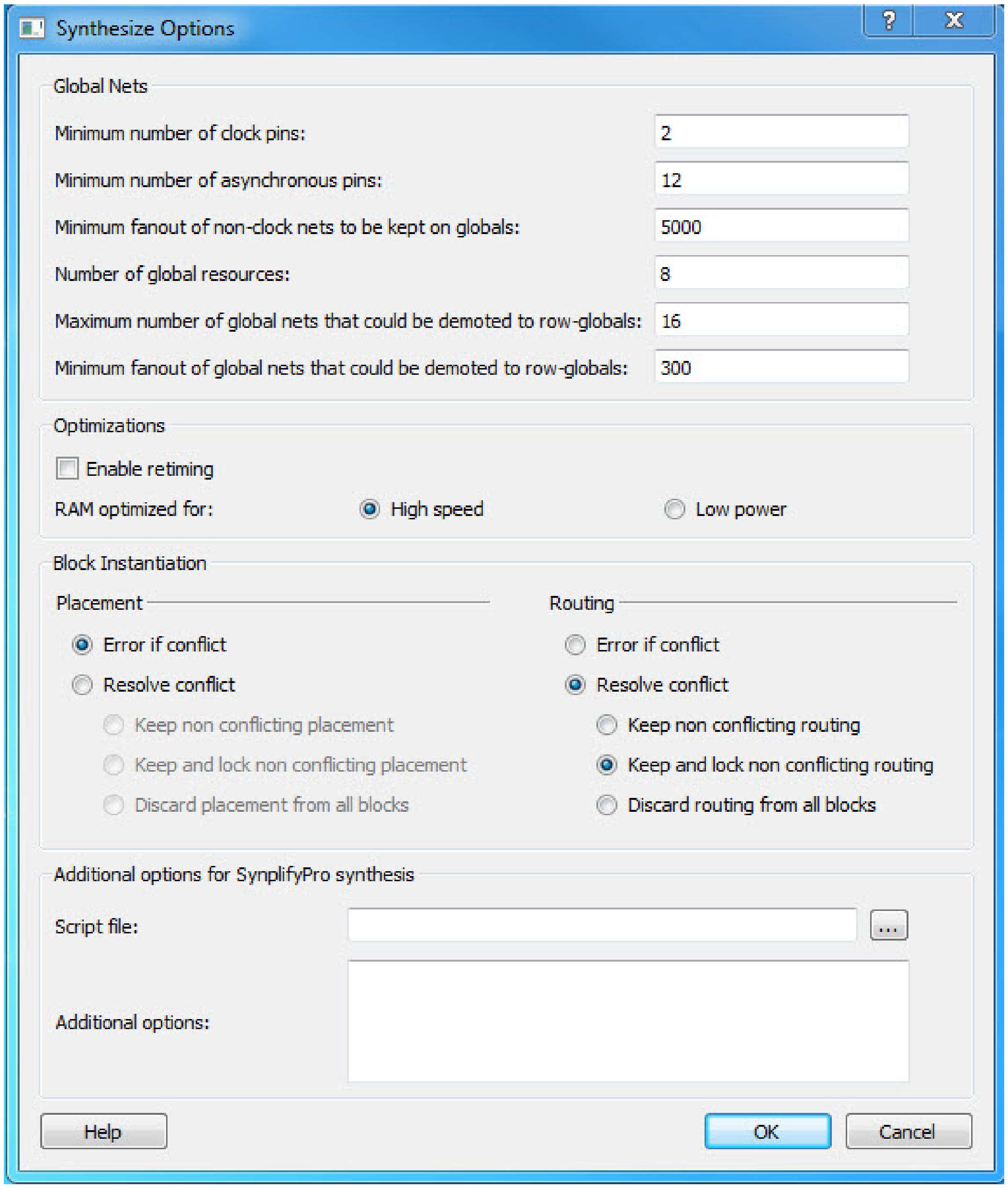
- **Keep non-conflicting placement** - If some instances get unplaced for any reason, the non-conflicting elements remaining are preserved but not locked (you can move them).
- **Keep and lock non-conflicting placement** - If some instances get unplaced for any reason, the remaining non-conflicting elements are preserved and locked.
- **Discard placement from all blocks** – Placement information will be discarded from all blocks even if there is no conflict.

Routing

Error if conflict - The Layout tool errors out if any preserved net routing in a designer block is deleted.

Resolve conflict

- **Keep non-conflicting routing**- If a nets' routing is removed for any reason, the routing for the non-conflicting nets is preserved but not locked (so that they can be rerouted). This is the default option.
- **Keep and lock non-conflicting routing**- If the routing is removed for any reason, the remaining non-conflicting nets are preserved and locked; they cannot be rerouted. This is the default option.
- **Discard routing from all blocks** – Routing information will be discarded from all blocks even if there is no conflict.


Synthesize Options

Global Nets

Minimum number of clock pins:	2
Minimum number of asynchronous pins:	12
Minimum fanout of non-clock nets to be kept on globals:	5000
Number of global resources:	8
Maximum number of global nets that could be demoted to row-globals:	16
Minimum fanout of global nets that could be demoted to row-globals:	300

Optimizations

☐ Enable retiming

RAM optimized for: [✓ High speed] Low power

Block Instantiation

Placement
- [✓ Error if conflict]
- Resolve conflict
  - Keep non conflicting placement
  - Keep and lock non conflicting placement
  - Discard placement from all blocks

Routing
- Error if conflict
- [✓ Resolve conflict]
  - Keep non conflicting routing
  - [✓ Keep and lock non conflicting routing]
  - Discard routing from all blocks

Additional options for SynplifyPro synthesis

Script file:

Additional options:

Figure 4 · Synthesis Options Dialog Box

Block and Block-related PDC Commands

move_block and set_block_options are two PDC commands available specifically for working with design blocks at the top level.

Use the move_block and set_block_options commands to make changes in your Top-Level design. See the respective help topics for more information.

In the top level design, put the move_block and set_block_options commands in an NDC file (**Design Flow Window > Manage Constraints > Open Manage Constraints View > Netlist Attributes > New > Create New Compile Netlist Constraints NDC**) and associate the NDC file with Synthesis.

define_region and assign_region are two PDC commands especially useful for floor planning. See their respective help topics for more information.

move_block

PDC command; moves a design block from its original, locked placement by preserving the relative placement between the instances. You can move the Block to the left, right, up, or down.

```
move_block -inst_name instance_name -up y -down y  -left x -right x  -non_logic value
```

Arguments

`-inst_name instance_name`

Specifies the name of the instance to move. If you do not know the name of the instance, run a Compile report or look at the names shown in the Block tab of the Chip Planner.

`-up y`

Moves the block up the specified number of rows. The value must be a positive integer.

`-down y`

Moves the block down the specified number of rows. The value must be a positive integer.

`-left x`

Moves the block left the specified number of columns. The value must be a positive integer.

`-right x`

Moves the block right the specified number of columns. The value must be a positive integer.

`-non_logic value`

Specifies what to do with the non-logic part of the block, if one exists. The following table shows the acceptable values for this argument:

Value	Description
move	Move the entire block.
keep	Move only the logic portion of the block (COMB/SEQ) and keep the rest locked in the same previous location, if there is no conflict with other blocks.
unplace	Move only the logic portion of the block (COMB/SEQ) and unplace the rest of it, such as I/Os and RAM.

Description

This command moves a block from its original, locked position to a new position.

You can move the entire block or just the logic part of it. You must use the -non_logic argument to specify what to do with the non-logic part of the block. You can find placement information about the block in the Block report. From the **Tools** menu in the designer software, choose **Reports > Block > Interface** to display the report that shows the location of the blocks.

The -up, -down, -left, and -right arguments enable you to specify how to move the block from its original placement. You cannot rotate the block, but the relative placement of macros within the block will be preserved and the placement will be locked. However, routing will be lost. You can either use the ChipPlanner tool or run a Block report to determine the location of the block.

The -non_logic argument enables you to move a block that includes non-logic instances, such as RAM or I/Os that are difficult to move. Once you have moved a part of a block, you can unplace the remaining parts of the block and then place them manually as necessary.

Note: Microsemi recommends that you move the block left or right by increments of 12. If not, placement may fail because it violates clustering constraints. Also, Microsemi recommends that you move the block up or down by increments of three.

Exceptions

- You must import this PDC command as a source file, not as an auxiliary file.
- You must use this PDC command if you want to preserve the relative placement and routing (if possible) of a block you are instantiating many times in your design. Only one instance will be preserved by default. To preserve other instances, you must move them using this command.

Examples

The following example moves the entire block (instance name instA) 12 columns to the right and 3 rows up::

```
move_block -inst_name instA -right 12 -up 3 -non_logic move
```

The following example moves only the logic portion of the block and unplaces the rest by 24 columns to the right and 6 rows up.

```
move_block -inst_name instA –right 24 –up 6 –non_logic unplace
```

See Also

set_block_options

set_block_options

PDC command; overrides the compile option for placement or routing conflicts for an instance of a block.

```
set_block_options -inst_name instance_name -placement_conflicts value -routing_conflicts
value
```

Arguments

```
-inst_name instance_name
```

Specifies the block instance name. If you do not know the name of the instance, run a Block Report (**Design > Reports > Blocks > Interface**) or look at the names shown in the Block tab of the Chip Planner.

```
-placement_conflicts value.
```

Specifies what to do when the software encounters a placement conflict. The following table shows the acceptable values for this argument:

Value	Description
error	Compile errors out if any instance from a Designer block becomes unplaced or its routing is deleted. This is the default compile option.

Value	Description
resolve	If some instances get unplaced for any reason, the non-conflicting elements remaining are also unplaced. Basically, if there are any conflicts, nothing from the block is kept.
keep	If some instances get unplaced for any reason, the non-conflicting elements remaining are preserved but not locked. Therefore, the placer can move them into another location if necessary.
lock	If some instances get unplaced for any reason, the non-conflicting elements remaining are preserved and locked.
discard	Discards any placement from the block, even if there are no conflicts.

`-routing_conflicts` *`value`*

Specifies what to do when the software encounters a routing conflict. The following table shows the acceptable values for this argument:

Value	Description
error	Compile errors out if any route in any preserved net from a Designer block is deleted.
resolve	If a route is removed from a net for any reason, the routing for the non-conflicting nets is also deleted. Basically, if there are any conflicts, no routes from the block are kept.
keep	If a route is removed from a net for any reason, the routing for the non-conflicting nets is kept unlocked. Therefore, the router can re-route these nets.
lock	If routing is removed from a net for any reason, the routing for the non-conflicting nets is kept as locked, and the router will not change them. This is the default compile option.
discard	Discards any routing from the block, even if there are no conflicts.

Description

This command enables you to override the compile option for placement or routing conflicts for an instance of a block.

Exceptions

You must import this PDC command as a source file, not as an auxiliary file.

If placement is discarded, the routing is automatically discarded too.

Examples

This example makes the designer software display an error if any instance from a block becomes unplaced or the routing is deleted:

```
set_block_options -inst_name instA -placement_conflicts ERROR -routing_conflicts ERROR
```

See Also

move_block

define_region

PDC command; defines either a rectangular region or a rectilinear region.

```
define_region -region_name <region_name> -type <inclusive|exclusive|empty> -x1 <integer> -y1
<integer> -x2 <integer> -y2 <integer> [-color <integer>] [-route <true|false>]
```

Note: The -color and -route parameters are optional.

Arguments

`-region_name` *`region_name`*

Specifies the region name. The name must be unique. Do not use reserved names such as “bank0” and “bank<N>” for region names. If the region cannot be created, the name is empty. A default name is generated if a name is not specified in this argument.

`-type <`*`inclusive`*`|`*`exclusive`*`|`*`empty`*`>`

Specifies the region type. The default is inclusive. The following table shows the acceptable values for this argument:

Region Type Value	Description
Empty	Empty regions cannot contain macros.
Exclusive	Only contains macros assigned to the region.
Inclusive	Can contain macros both assigned and unassigned to the region.

`-x1 -y1 -x2 -y2`

Specifies the series of coordinate pairs that constitute the region. These rectangles may or may not overlap. They are given as x1 y1 x2 y2 (where x1, y1 is the lower left and x2 y2 is the upper right corner in row/column coordinates). You must specify at least one set of coordinates.

`-color` *`value`*

Specifies the color of the region. The following table shows the recommended values for this argument:

Color	Decimal Value
🟦	16776960
🟩	65280
🟦	16711680
🟦	16760960
🟥	255
🟪	16711935
🟨	65535

Color	Decimal Value
🟧	33023
🟥	8421631
🟩	9568200
🟪	8323199
⬜	12632256

`-route` *`value`*

Specifies whether to direct the routing of all nets internal to a region to be constrained within that region. A net is internal to a region if its source and destination pins are assigned to the region. You can enter one of the following values:

Constrain Routing Value	Description
true	Constrain the routing of nets within the region as well as the placement.
false	Do not constrain the routing of nets within the region. Only constrain the placement. This is the default value.

Note: Local clocks and global clocks are excluded from the -route option. Also, interface nets are excluded from the –route option because they cross region boundaries.

An empty routing region is an empty placement region. If -route is "yes", no routing is allowed inside the empty region. However, local clocks and globals can cross empty regions.

An exclusive routing region is an exclusive placement region (rectilinear area with assigned macros) along with the following additional constraints:

- For all nets internal to the region (the source and all destinations belong to the region), routing must be inside the region (that is, such nets cannot be assigned any routing resource which is outside the region or crosses the region boundaries).
- Nets without pins inside the region cannot be assigned any routing resource which is inside the region or crosses any region boundaries.

An inclusive routing region is an inclusive placement region (rectilinear area with assigned macros) along with the following additional constraints:

- For all nets internal to the region (the source and all destinations belong to the region), routing must be inside the region (that is, such nets cannot be assigned any routing resource which is outside the region or crosses the region boundaries).
- Nets not internal to the region can be assigned routing resources within the region.

Description

Unlocked macros in empty or exclusive regions are unassigned from that region. You cannot create empty regions in areas that contain locked macros.

Use inclusive or exclusive region constraints if you intend to assign logic to a region. An inclusive region constraint with no macros assigned to it has no effect. An exclusive region constraint with no macros assigned to it is equivalent to an empty region.

Note: If macros assigned to a region exceed the area's capacity, the region's Properties Window displays the overbooked resources (over 100 percent resource utilization) in red.

Examples

The following example defines an empty rectangular region called UserRegion1 with lower-left co-ordinates (100,46) and upper-right co-ordinates (102,50).

```
define_region -region_name UserRegion1 -type empty -x1 100 -y1 46 -x2 102 -y2 50
```

The following example defines an inclusive rectilinear region with the name UserRegion2. This region contains two rectangular areas, one with lower-left co-ordinates (12,39) and upper-right co-ordinates (23,41) and another rectangle with lower-left co-ordinates (12,33) and upper-right co-ordinates (23,35).

```
define_region -region_name UserRegion2 -type exclusive -x1 12 -y1 39 -x2 23 -y2 41 -x1 12
-y1 33\
-x2 23 -y2 35
```

The following examples define three regions with three different colors:

```
define_region -region_name UserRegion0 -color 128 -x1 50 -y1 19 -x2 60 -y2 25
define_region -region_name UserRegion1 -color 16711935 -x1 11 -y1 2 -x2 55 -y2 29
define_region -region_name UserRegion2 -color 8388736 -x1 61 -y1 6 -x2 69 -y2 19
```

See Also

assign_region

assign_region

PDC command; constrains a set of macros to a specified region.

```
assign_region -region_name region_name -inst_name macro_name+
```

Arguments

region_name

Specifies the region to which the macros are assigned. The macros are constrained to this region. Because the define_region command returns a region object, you can write a simpler command such as assign_region [define_region]+ [macro_name]+.

macro_name

Specifies the macro(s) to assign to the region. You must specify at least one macro name. You can use the following wildcard characters in macro names:

Wildcard	What It Does
\	Interprets the next character as a non-special character
?	Matches any single character
*	Matches any string

The region must be created before you can assign macros to it. If the region creation PDC command and the macro assignment command are in different PDC files, the order of the PDC files is important.

You can assign only hard macros or their instances to a region. You cannot assign a group name. A hard macro is a logic cell consisting of one or more silicon modules with locked relative placement.

The macro name must be a name with full hierarchical path.

Notes:

- The region must be created before you can assign macros to it. If the region creation PDC command and the macro assignment command are in different PDC files, the order of the PDC files is important.
- You can assign only hard macros or their instances to a region. You cannot assign a group name. A hard macro is a logic cell consisting of one or more silicon modules with locked relative placement.
- The macro name must be a name with full hierarchical path.

Examples

In the following example, two macros are assigned to a region:

```
assign_region -region_name UserRegion1 -inst_name “test_0/AND2_0 test_0/AND2_1”
```

In the following example, all macros whose names have the prefix des01/Counter_1 (or all macros whose names match the expression des01/Counter_1/*) are assigned to a region:

```
assign_region -region_name User_region2 -inst_name des01/Counter_1/*
```

Publish Block - Configuration Options

To view this dialog box you must first Enable Block Creation in the **Libero SoC Project Settings** or **New Project Creation Wizard**. After Block Creation is enabled Publish Block appears in the Design Flow window. Expand **Publish Design**, right-click **Publish Block** and choose **Export.**

Publish Block Configuration

Publish Placement- Check this box to publish the placement information for the Block. Note that you must assign all macros to regions or lock them in order to Publish Placement.

If checked, the published Block can only be instantiated and used in a top level design with the same family and device. If the Block contains I/Os, the published Block can only be instantiated and used in a top level design with the same family, device and package.

If unchecked, only a netlist is published for the block. The published block can be instantiated and used in a top level design for any device and package in the same device family as the block.

Publish Routing - Check this box to retain the routing information with the block when published.

Publish Region - Check this box to retain the region constraint information with the block when published.

Language

Select your Block Hardware Description Language (Verilog or VHDL). The default is the Preferred HDL type set in your **Project Settings**.